IPDPS 2009: 23rd IEEE International Parallel & Distributed Processing Symposium

Paride Dagna
CILEA, Segrate

Abstract

IPDPS costituisce un importante evento internazionale dedicato alla divulgazione e promozione dei molteplici aspetti del calcolo e dei processi paralleli e distribuiti. I temi trattati nell'edizione di Roma 2009, erano accomunati dal notevole interesse della comunità scientifica verso il calcolo ad alte prestazioni eseguito sulle GPU, ovvero le Graphical Processing Unit, di cui sono costituite le schede grafiche. Questo tipo di tecnologia sembra essere molto promettente, poiché le GPU sono intrinsecamente e massicciamente parallele e pertanto in grado di fornire singolarmente la potenza di calcolo di svariate CPU.

IPDPS is an important international event for engineers and scientists to present their latest research findings in the fields of parallel processing and distributed computing. The themes of IPDPS 2009 were tied together by the great interest of the scientific community for the GPU computing. This kind of technology seems very promising because GPUs are intrinsically and massively parallel with a consequent tremendous computing power if compared to a modern single CPU.

Keywords: Gpu, Cuda, calcolo parallelo, applicazioni, algoritmi, software.

Fig. 1 – Logo della conferenza IPDPS 2009

IPDPS 2009 – Roma

IPDPS 2009, la ventitreesima nella cronologia delle edizioni di questo evento di importanza internazionale, si è svolta a Roma dal 25 al 29 Maggio, al Grand Hotel Palazzo Carpegna.

La conferenza è stata organizzata dall'IEEE Computer Society Technical Committee on Parallel Processing, in collaborazione con l'IEEE Computer Society Technical Committee on Computer Architecture, l'IEEE Computer Society Technical Committee on Distributed Processing ed il dipartimento di Informatica dell'Università La Sapienza di Roma.

Le conferenze IPDPS sono organizzate annualmente da un comitato di rappresentanti internazionali, che curano il buon andamento delle manifestazioni e decidono di anno in anno le date ed i luoghi di tali eventi.
L'edizione del prossimo anno si terrà ad Atlanta, Georgia, USA.

Data la vastità e l'eterogeneità degli argomenti trattati, il programma degli interventi è stato strutturato secondo alcuni temi generali, inseriti all'interno delle voci "Algoritmi", "Applicazioni", "Architetture" e "Software", presentati in 21 workshop e 25 sessioni parallele.

I temi principali

Nelle accoglienti aule del Grand Hotel Palazzo Carpegna la comunità scientifica internazionale ha presentato le proprie ricerche e si è confrontata sui risultati e sulle possibili evoluzioni delle tematiche riguardanti il vasto ed eterogeneo mondo dell'high performance computing.

Durante le cinque giornate di IPDPS 2009 sono emerse le enormi potenzialità computazionali nascoste all'interno di questo nuovo settore dell'HPC. Allo stesso modo si sono però

evidenziate le incertezze e gli ostacoli che dovranno essere superati per permettere al GPU computing di diventare una tecnologia matura.

La giornata di apertura del convegno è stata caratterizzata da una serie di workshop intesi a fornire una visione generale sullo stato dell'arte nell'ambito degli strumenti, quali compilatori, debugger e analizzatori di performance, per il calcolo parallelo e distribuito, il cluster ed il grid computing.

Già dai primi interventi è risultato evidente che il filo conduttore di tutte le giornate sarebbe stato l'enorme interesse suscitato dal recente avvento dei sistemi eterogenei multicore, composti da CPU e GPU.

A tal proposito sono state presentate le linee di ricerca nell'area delle applicazioni, dei modelli computazionali, dei linguaggi, dei compilatori e dei tools per il controllo, la programmazione e la valutazione delle performance di tali sistemi.

E' stato sottolineato come diventi indispensabile la disponibilità [1] di strumenti di tuning automatici, dal momento che il processo di ottimizzazione degli applicativi risulta sempre più complesso e legato alle differenti soluzioni di hardware e sistemi operativi disponibili.

E' stato dimostrato che impiegando questi mezzi [2], [3] è stato possibile incrementare la velocità di esecuzione dei codici da 1.4 a 3.6 volte rispetto ai tempi ottenuti dalla semplice compilazione effettuata con gli usuali compilatori commerciali disponibili.

La continua crescente richiesta di calcolo, non potrà essere ancora supportata a lungo [4], dalle soluzioni cluster tradizionali, caratterizzate da sistemi composti da migliaia di nodi e cpu, a causa dell'eccessiva richiesta di energia per il loro mantenimento e raffiedamento.

Inoltre su tali sistemi la scalabilità degli applicativi è limitata dalla capacità dello scambio di informazioni fra i nodi data dai limiti della rete di interconnessione.

Le moderne GPU che compongono le schede grafiche NVIDIA di alta gamma, appartenenti alla famiglia GeForce, Quadro e Tesla, le ATI Radeon e l'imminente processore Intel Larrabee, risultano intrinsecamente e massicciamente parallele, essendo costituite al loro interno da molti cores computazionali dove migliaia di threads eseguono innumerevoli operazioni contemporaneamente.

Utilizzando questi componenti e compilando i codici in modo da sfruttarne le caratteristiche architetturali si possono ottenere speed-up da 10 a 100 volte[5] rispetto alle prestazioni date dai processori tradizionali.

Osservando la roadmap mostrata dall'industria dei semiconduttori, entro la fine del 2016, questi processori potranno scalare con uno speed-up di oltre 1000 rispetto al singolo core.

Un'intera sessione del convegno è stata dedicata alla biologia computazionale nell'ambito del GPU computing.

Questo settore applicativo è in grado di ottenere rilevanti benefici con l'adozione di questo hardware specializzato.

John Paul Walters [6] ed il suo gruppo di ricerca del Department of Computer Science and Engineering dell'University di Buffalo hanno mostrato come l'applicativo di segmentazione "Markov random fields based" (MRF) e l'algoritmo HMMER's Viterbi [7], ottengano rispettivamente uno speed-up di 130X e 38X se eseguiti su scheda grafica NVIDIA 8800 GTX Ultra (Fig. 2 e Fig.3).

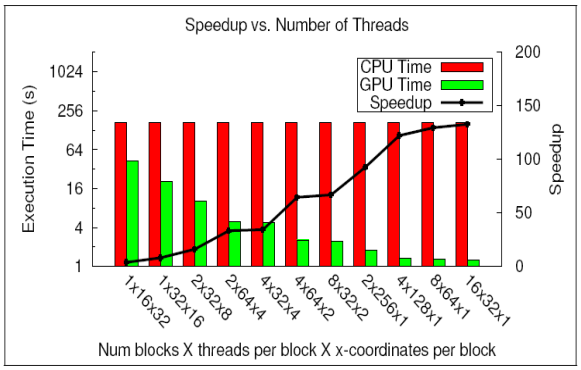

Fig. 2 – Speed-up del codice MRF GPU vs CPU.

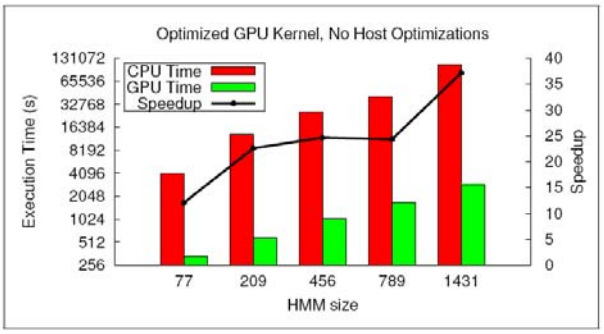

Fig. 3 – Speed-up dell' algoritmo HMMER's GPU vs CPU.

Altre applicazioni nel campo della bioinformatica come l'analisi e l'allineamento di sequenze genomiche e proteiche [8], realizzate utilizzando GPU con codici che utilizzavano l'algoritmo di Smith-Waterman, hanno ottenuto

un incremento di 23 volte rispetto all'esecuzione su un singolo processore.

Le GPU che compongono le schede grafiche sono co-processori dotati di un'enorme capacità di calcolo che ben si adattano alla soluzione di calcoli matriciali ed in generale di algebra linare; è stato mostrato [9], come la soluzione SVD (singular value decomposition) di una matrice densa e la diagonalizzazione, se effettuata su GPU NVIDIA GTX 280, presenti un incremento di performance di 60 volte rispetto al software Matlab e 8 volte rispetto all' Intel MKL eseguiti su processore Intel Dual Core 2.66 GHz (Fig. 4).

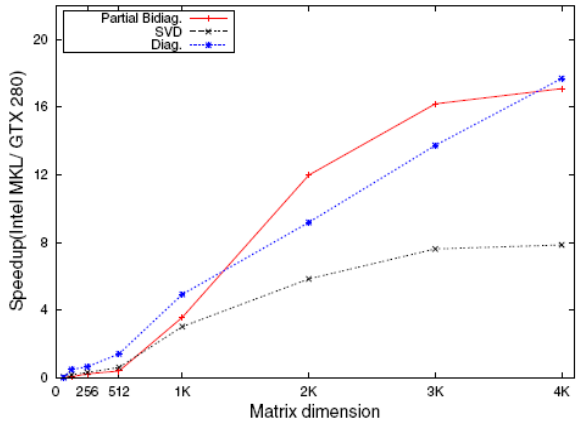

Fig. 4 – Speed-up rispetto all'Intel MKL dell'algoritmo SVD e della diagonalizzazione matriciale eseguita su GPU su sched NVIDIA GTX 280.

E' tuttavia fondamentale ricordare che in realtà il processo di porting del codice da CPU a GPU non è quasi mai immediato.

Il linguaggio di programmazione per GPU maggiormente utilizzato è CUDA.

Quest'ultimo, sviluppato dalla società NVIDIA, può essere visto come una libreria di funzioni e direttive che costituiscono un'estensione del "C.

L'utilizzo di tale strumento non è banale e per ottenere le performance indicate negli esempi precedenti è necessaria una notevole conoscenza delle caratteristiche dell'hardware del co-processore.

E' fondamentale ottimizzare ed allineare l'accesso dei threads ai diversi strati di memoria di cui è composta la GPU e ridurre il più possibile gli scambi di dati fra GPU e CPU-RAM, dal momento che questa è l'operazione di gran lunga più lenta, richiedendo circa 400 cicli di clock.

Per questo motivo i ricercatori e le case produttrici di software stanno cercando di ideare degli strumenti che permettano di semplificare e automatizzare il processo di porting dell'applicativo da CPU a GPU.

Uno di questi strumenti, sviluppato presso il Research Group Programming Languages dell'Università di Kassel in Germania [10], è il framework CuPP [11], che ha mostrato delle interessanti proprietà..

Con CuPP il programmatore dispone di una serie di interfacce di alto livello che permettono di gestire la GPU, le chiamate alle funzioni per il co-processore e la sua memoria, oltre a fornire supporto per classi e trutture dati.

Il processo di trasformazione del codice per GPU risulta così più efficiente, inoltre l'integrazione nel codice delle chiamate alle funzioni CuPP ha il pregio di causare un overhead molto limitato (1%), non influendo sulle prestazioni globali dell'applicativo.

Un altro strumento interessante è G-Adapt [12], un framework adattativo, capace di trovare la configurazione ottimale del codice in base all'input.

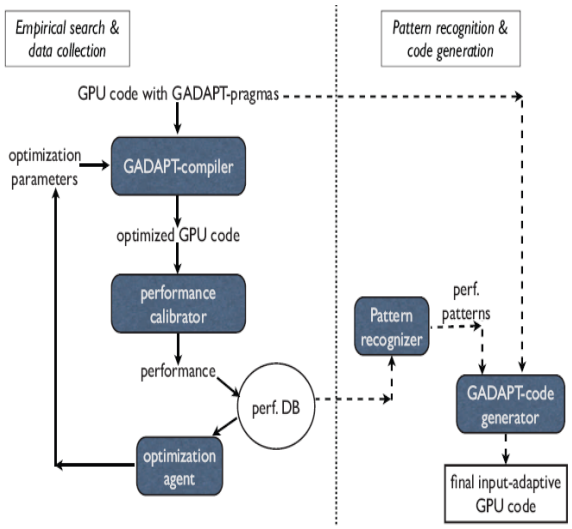

Fig. 5 – Schema rappresentativo del processo di ottimizzazione utilizzato da G-Adapt.

Il processo adattativo avviene in due fasi (Fig. 5).

Inizialmente, il codice contentente le direttive di G-Adapt, dopo la compilazione, subisce una serie di trasformazioni per la ricerca empirica di parametri e combinazioni ottimali che vengono memorizzati in un database. Terminata questa fase, il database viene utilizzato per la ricerca delle relazioni migliori

fra i parametri memorizzati e gli input del codice. Quindi G-Adapt trasforma il codice originale per GPU in un applicativo in grado di selezionare automaticamente la versione più performante a seguito di un dato input.

Le performance sono state valutate utilizzando come casi di test gli esempi contenuti nella distribuzione ufficiale di CUDA. Rispetto ai sorgenti compilati con il compilatore CUDA nvcc [13], quelli ottimizzati con G-Adapt hanno mostrato uno speed-up di circa 6X sull'operazione di riduzione e di circa 2X su operazioni algebriche molto importanti quali convoluzione, moltiplicazione matriciale e prodotto scalare (Fig. 6).

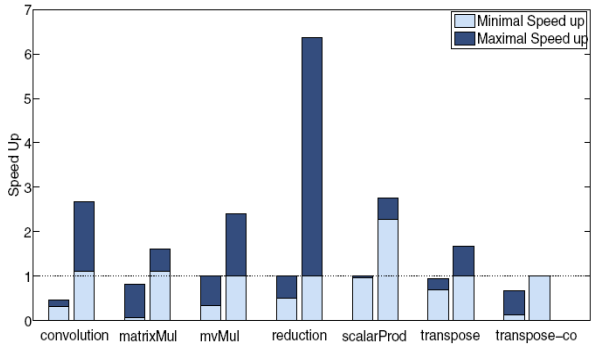

Fig. 6 – Speed-up ottenuto attraverso il software G-Adapt rispetto ad alcuni casi di test compilati con il compilatore CUDA nvcc.

Conclusioni

L'obiettivo ed il desiderio della comunità scientifica, che è trasparso all'interno dei numerosi interventi, è stato quello di poter disporre in un tempo relativamente breve, di un ambiente di sviluppo completo grazie al quale il programmatore potrà scrivere codici paralleli in grado di sfruttare pienamente le potenzialità offerte da cluster composti da soluzioni miste CPU-GPU, dove le direttive MPI e le funzioni per i co-processori si fondono in un unico linguaggio di alto livello.

Bibliografia

[1] Ananta Tiwari, Chun Chen, Jacqueline Chame, "A Scalable Auto-tuning Framework for Compiler Optimization", *IPDPS 2009: Proceedings*.

[2] Active Harmony URL: http://www.dyninst.org/harmony/

[3] Chill. URL: http://www.cs.utah.edu/~chunchen/chill/

[4] Leonid Oliker, "Green Flash: Designing an Energy Efficient Climate Supercomputer", *IPDPS 2009: Proceedings*.

[5] Wen-mei W. Hwu, "Many-core Parallel Computing - Can Compilers and Tools Do the Heavy Lifting?", *IPDPS 2009: Proceedings*.

[6] John Paul Walters, Vidyananth Balu, Suryaprakash Kompalli, Vipin Chaudhary, "Evaluating the use of GPUs in Liver Image Segmentation and HMMER Database Searches", *IPDPS 2009: Proceedings*.

[7] HMMER. URL: http://hmmer.janelia.org/

[8] Gregory M. Striemer, "Sequence Alignment with GPU: Performance and Design Challenges", *IPDPS 2009: Proceedings*.

[9] Sheetal Lahabar, P. J. Narayanan, "Singular Value Decomposition on GPU using CUDA", *IPDPS 2009: Proceedings*.

[10] Jens Breitbart, "CuPP – A framework for easy CUDA integration", *IPDPS 2009: Proceedings*.

[11] CuPP. URL: http://www.plm.eecs.uni-kassel.de/CuPP/

[12] Yixun Liu, Eddy Z. Zhang, Xipeng Shen, "A Cross-Input Adaptive Framework for GPU Program Optimizations", *IPDPS 2009: Proceedings*.

[13] NVCC. URL: http://www.nvidia.com/object/cuda_get.html